AMENDMENTS TO THE SPECIFICATION

Page 6, lines 9-19:

As each sample period corresponds to the length of two symbols, there are potentially two transmitted symbols in the captured sample. However, the sample will not be synchronised to the start of a particular symbol. So, as shown for example in Figure 3(a), at time t0 the last portion of a symbol associated with the (n-1)th symbol of the first multi-path of the first caller is received. The nth symbol Sn of the first multi-path of the first caller is then received in full. Then finally the first part of the (n+1)th symbol of the first multi-path of the first caller is received. Thus the first sample written into the memory in the first processing period include two partial symbols and one full symbol of the first multi-path of the first caller. ~~Similalrly~~ Similarly for other multi-paths in the first processing period, full and partial symbols are stored in the memory, as exemplified by Figures 3(b) to 3(d).

Page 7, lines 10-22:

The possible implementation of the timing error detection and estimation circuitry 14 will be apparent to one skilled in the art. In operation, the address control circuitry 12 reads stored sample data from the memory, and presents it on lines 28 to the fine timing adjustment circuitry 4. The address control circuitry controls the sample memory to provide the full two-symbol period sample of Figure 3(a) to the fine timing adjustment circuitry. The two symbol period sample ~~conatins~~ contains, as discussed hereinabove, various multi-paths for various callers as shown by way of example in the first processing period of Figures 3(a) to 3(d).

The finger memory 18 is controlled by the control circuit 19 to switch between contexts in accordance with the multi-path currently being processed. The fine timing adjustment circuitry uses the fractional error timing information generated by the timing error detection and estimation ~~circuiry~~ circuitry and

stored in the finger memory 18 to fractionally adjust the timing of each multi-path in the given processing period.

Page 8, lines 17-24:

The code generator 64 receives the timing signals from the finger memory 18 on line 44, and receives and transmits the signals on lines 48 and 50 between the de-spreading circuitry 4 and the finger memory 18. The code generator is loaded, from finger memory 18 with the spreading code known to be used by the first caller. The finger memory stores the spreading codes for the different callers. Thus each context in the finger memory 18 will additionally be associated with a spreading code. Under the control of the control circuit 19, the finger memory outputs the ~~appropriatye~~ appropriate spreading code for the current context on line 48.

Page 9, lines 17-21:

As, in general, a symbol boundary will not coincide with the end of a processing period the content of the summer 62 is stored in the finger memory at the end of the partial symbol using lines 54, and then reinstated when that multi-path is to be next processed using lines 52. The partial summed ~~reslut~~ result for a partial symbol is stored in the finger memory in association with the context associated with that multi-path.

Page 10, lines 23-28:

Within the first processing period, the other symbols present for different multi-paths and ~~differencallers~~ different callers are processed in a similar way. Referring to the example of Figures 3(a) to 3(d), at the end of the first processing ~~p~~period the finger memory will additionally include partial de-spreading results associated with the second multi-path of the (n+1)th symbol of the first caller, the first multi-path of the (n+1)th symbol of the second caller, and the second multi-path of the (n+1)th symbol of the second caller.

Page 14, lines 13-21:

Where time is required for other processing tasks, that time must be subtracted from the processing clock rate prior to performing the above calculation. For example if time is required for the reading of the sample RAM for synchronisation calculations, the clock rate may be scaled given the overhead of one buffer read per processing period. Hence the clock rate may be scaled to:

~~$$fp = fp - \max\left(\frac{N - Ns}{N}\right)$$~~

$$fp = fp - \max\left(\frac{N - Ns}{N}\right)$$

Using the values from the above example and Ns=14, a result of fp=94.53MHz is obtained. The number of supportable fingers is then 21.